مقدمه

هدف اصلی این گزارش، بررسی اجمالی مبحث عاملهای هوشمند یادگیر است. از آنجا که تعریف متحدی برای مفهوم عامل وجود ندارد، فعالیت‌های بسیار زیادی با نام عاملها در حال انجام است. در این گزارش سعی بر آن است که تا حدی مفهوم عامل بررسی شده و حدود و ثغور فعالیت‌های انجام شده در آن، با تأکید بر عاملهای هوشمند یادگیر، مورد بررسی قرار گیرد.

بدین منظور در بخش ۱ مفهوم عامل مورد بررسی قرار گرفته و تعاریف و معماری‌های شاخص آن مورد بررسی قرار گرفته است. در بخش ۲ نیز مفهوم یادگیری، سرچشمه‌های الگوریتمهای یادگیری و ضرورت یادگیری در سیستم‌ها بیان شده است. در بخش سوم عامل هوشمند یادگیر تعریف شده است و به معماری‌های برجسته آن و روشهای متداول یادگیری در آن پرداخته شده است و در بخش پایانی نیز کاربردهایی از عاملهای هوشمند بیان شده است.

۱-۱- مقدمه

عامل نرم‌افزاری، ریشه در سیستم‌های چندعاملی دارد؛ که این سیستم‌ها نیز به نوبهٔ خود یکی از سه مبحث متداول در هوش مصنوعی توزیع شده، یعنی، هوش مصنوعی موازی و حل مسألهٔ توزیع شده و سیستمهای چندعاملی می باشد [۲۱]. بنابراین بدیهی است که قابلیتهای سیستم‌های هوش مصنوعی توزیع شده را به ارث برد. برای مثال، ماژولاریتی، سرعت و قابلیت اطمینان بالا که محصول توزیع شدگی محاسبات است؛ و قابلیتهایی مانند پردازش در سطح دانش، نگهداری آسانتر و استفاده مجدد که محصول هوش مصنوعی است را به ارث می برد.

برای بررسی عاملها به سه مبحث باید پرداخته شود[۳۱].

۱- **نظریهٔ عاملها**: که دلالت بر چیستی آنها و فرمالیزمهای ریاضی جهت استنتاج و نمایش خواص عاملها می کند.

۲- **معماری عاملها**: مدلهای مهندسی نرم‌افزاری هستند، که با توجه به خواص مشخص شده توسط نظریه پردازها برای عاملها بوجود می آیند.

۳- **زبان عاملها**: سیستم‌های نرم‌افزاری برای برنامه‌نویسی و کار با عاملها هستند. این زبانها باید اصولی را که نظریه پردازها بیان می‌کنند، پیاده سازی کنند.

۱-۲- معرفی عامل

پیدایش مفهوم عامل را می‌توان اوائل دهۀ ۱۹۷۰ دانست. در آن زمان Hewitt مدل "اشیای همروند" را مطرح کرد[۱۰]. در مدل وی مفهوم اشیایی خودگردان[۱]، تعاملی[۲] و با قابلیت اجرای همروند عرضه شد، که او آنها را "هنرپیشه[۳]" نامید. هر شیء دارای حالات درونی خود بوده و ارتباط آن با اشیای دیگر از طریق ارسال پیام صورت می گرفت. به گفتهٔ او یک هنرپیشه عاملی محاسباتی است، که دارای رفتار و آدرس است. عاملها با ارسال پیام با همدیگر ارتباط برقرار نموده و کنشهایشان را بصورت همروند انجام می‌دهند.

۱-۲-۱- تعریف عامل

تعریف واحدی از مفهوم عامل موجود نمی باشد و هر یک از صاحبنظران این موضوع با توجه به دیدگاه خود تعریفی را ارائه می دهند.. به همین دلیل تعاریف مهم عامل از دیدگاه دانشمندان و محققان معروف در این بخش آمده است.

Russell وNorvig: عامل هر چیزی است که محیط خود را باحسگرها درک می کند و در محیط با کنشگرها تاثیر می گذارد. این تعریف بسیار وابسته به محیط است[۲۳].

Maes: عامل سیستمی محاسباتی است،که در محیطی پویا و پیچیده سکنی دارد. بصورت خودگردان محیط را حس کرده و با توجه به اهداف خود کنش یا کنشهایی را صورت می دهد[۱۸].

Pattie Maes یکی از پیشتازان تحقیقات عاملهاست. او در تعریف عامل تأکید دارد که عامل باید قادر باشد به تنهایی کار کند تا بتواند اهدافی را بر آورده سازد. همچنین محیط عامل را به محیطی پیچیده و پویا محدود می داند.

---

[۱] Autonomous

[۲] Interactive

[۳] Actor

Smith, Cypher وSpohrer : عامل موجودیت نرم افزاریِ مانایی[1] است، که برای هدفی خاص طراحی شده است. مانایی عامل آن را از سابروتین تمیز می دهد و خاص منظوره بودن آن را از کل کاربرد متمایز می سازد[۲۸]. در تعریف فوق مانایی عاملها جنبهٔ جدیدی است که به آنها افزوده شده است.

Barbara Hayes-Roth: از دید او عامل، نرم افزاری است که دائماً سه عمل زیر را تکرار می کند[۹]:

۱- درک شرایط پویای محیط

۲- فعالیت جهت تاثیر گذاری بر محیط

۳- استنتاج به منظور تعبیر یافته ها ، حل مسائل ، حصول منتجات و مشخص کردن فعالیتها

Roth براین نکته اصرار دارد که عامل باید درفرآیند انتخاب کنش ، استنتاج کند.

Wooldridgeو Jennings : عامل سیستمی نرم افزاری یا سخت افزاری است که خواص زیر را دارا باشد[۱٤]:

۱- خودگردانی[2] : عامل بدون دخالت مستقیم انسانها یا دیگران انجام وظیفه می نماید و بر کنشها و وضعیت داخلی خود نوعی کنترل دارد.

۲- قابلیتهای اجتماعی[3]: عامل با سایر عاملها (و یا انسانها) از طریق نوعی زبان ارتباط برقرار می کند.

۳- قابلیت واکنش[4] : عامل محیط اطراف خود را درک کرده (که این محیط ممکن است: دنیای فیزیکی، کاربر با واسط کاربر ، مجموعه ای از سایر عوامل ، اینترنت یا احتمالا مجموعه ای از همهٔ اینها) و در زمان مناسب به تغییرات رخ داده در آن پاسخ می دهد.

٤- کنش گرایی[5]: عاملها نه تنها به محیط پاسخ می دهند، بلکه با رفتاری هدف گرا هدفی را دنبال می کنند.

Michael Coen: عامل نرم افزاری برنامه ای است که در مکالمه ، مذاکره و هماهنگی جهت انتقال اطلاعات به کار می رود[۲۱].

Brustoloni و Franklin: عامل سیستمی خودگردان، و دارای کنش هدف گرا در جهان واقع می باشد[۳].

۱-۲-۲- تفاوتهای هوش مصنوعی کلاسیک و عاملها

Maes این تفاوتها را بصورت زیر بیان می‌دارد[۱۷]:

۱- هوش مصنوعی کلاسیک بر روی مسائلی تمرکز دارد که ایزوله شده‌اند و غالبا بطور عمیق کارشناسی شده‌اند. بنابراین، سیستم‌های هوش مصنوعی به طور کلی به صورت عمیق و حرفه‌ای تنها با یک مسأله برخورد می‌کنند. حال آنکه عامل باید روی چندین مسأله در تخصصهای گوناگون، کار کند. معمولاً این تخصصها کم عمق‌تر می‌باشند. برای یک روبوت این تخصصها می‌توانند حرکت، ناوبری، شارژ نگه داشتن باتری‌ها، جمع‌آوری اشیا و غیره باشد.

---

[1] Persistent

[2] Autonomy

[3] Social Ability

[4] Reactivity

[5] Proactiveness

۲- هوش مصنوعی کلاسیک روی سیستم‌های بسته که تعامل مستقیم با محیط مسأله ندارند، کار می کند.. ارتباط سیستم با محیط به طور غیرمستقیم بوده و از طریق کاربر برقرار می‌شود. کاربر مشکلی را در دامنه تشخیص داده، سپس آنرا به زبان سمبولیک قابل فهم برای سیستم بیان می‌کنند؛ سیستم با توجه به توصیف مسأله، جواب را بصورت سمبولیک ارائه می‌دهد؛ پس از آن، وظیفهٔ کاربراست که جواب را درک کرده و در محیط پیاده‌سازی کند. در مقابل ، عامل یک سیستم باز است. یعنی، عامل در محیط قرار گرفته و با حسگرها و کنشگرها مستقیماً با دامنهٔ مسألهٔ خود ارتباط دارد و می‌تواند دامنهٔ خود را به کمک این کنشگرها متأثر سازد. دامنهٔ مسأله معمولاً بسیار پویا است. بدین معنا که سیستم، زمان محدودی برای عمل دارد وامکان وقوع رخداد غیرمنتظره وجود دارد و معمولاً با دیگر عوامل فعال در محیط، تعامل دارد.

۳- اکثر سیستم‌های هوش مصنوعی قدیمی تنها به حل یک مسأله می‌پردازند. مسأله‌ای را سیستم باید حل کند که توسط کاربر انسانی به سیستم می‌دهد. اغلب، این گونه سیستم‌ها محدودیت زمانی برای حل مسأله ندارند و مسؤولیت مقابله با وقفه‌ها وظیفهٔ آنها نیست. از دید اینگونه سیستم‌ها، دامنه و مسأله درطول زمان حل مسأله تغییر نمی‌کند. در مقابل عامل، سیستم کاملاً خودگردان بوده و باید محیط را تحت نظر داشته باشد و خود بفهمد که مسأله بعدی با هدف مورد انتظار بعدی چیست؟ علاوه بر آن باید زمان را در نظر گرفته و با تعداد زیادی هدف گوناگون و گاه متناقض کار کند.

٤- در نهایت، هوش مصنوعی کلاسیک، معمولاً به جنبه‌های توسعهٔ سیستم، اهمیتی نمی‌دهد. مثلاً اینکه چگونه ساختارهای دانش بوجود می‌آید و چگونه آنها باید در طول زمان تغییر کنند. آنها نیازی به منعطف بودن نسبت به تغییر حالات ندارند . اما در مقابل، عاملها تأکید زیادی بر روی انعطاف و یادگیری دارند. این بدین معنی است که سیستم در طول زمان ساختارهای درونی خود را با توجه به تجربیات کسب شده از محیط، بهبود می‌بخشد. عامل، مداوماً ساختارهای خود را به کمک روشهای یادگیری افزایشی یا استقرایی بروز می‌رساند.

۱-۲-۳- نظریهٔ عامل

در بخشهای قبل مروری اجمالی بر مفهوم عامل داشتیم. در این بخش به تئوری عامل می‌پردازیم. منظور از تئوری عامل فرمالیزمهایی است که خصیصه‌های عامل را بیان می‌دارد.

بحث را از Seel شروع می‌کنیم که می‌گوید "عامل موجودیتی است که دارای اعتقاد، آرزو، اراده و... است"[۲٤]. بر اساس نظر وی، Dennett مفهوم سیستمهای ارادی را پدید آورد[۵]. سیستم‌های ارادی، سیستم‌هایی هستند که رفتار آنها با اندازه‌گیری اعتقاد و آرزو بدست می‌آید.Dennett سیستم‌های ارادی را در دو سطح بیان می‌دارد:

۱- سیستم ارادی مرتبهٔ اول: دارای اعتقاد و آرزو است امّا اعتقاد و آرزویی در مورد اعتقادات و آرزوها وجود ندارد.

۲- سیستم ارادی مرتبهٔ دوم: پیچیده‌تر از قبلی بوده و اعتقاد و آرزوهای خود نیز دارای اعتقاد و آرزو هستند.

سؤال مهم در این مورد این است که" آیا خصیصه‌بندی عقاید و آرزوها درست و یا دارای فایده است یا خیر؟". McCarthy در پاسخ به این سؤال را این گونه می دهد[۱۹]. این خصیصه‌بندی زمانی درست است که توصیفهایی که برای ماشین استفاده می‌کنیم حاوی همان اطلاعاتی باشد که در مورد انسان صادق است و زمانی

مفید است که توصیف بتواند به فهم ما در مورد ساختار ماشین، رفتار آینده یا گذشتهٔ آن یا چگونگی تعمیر یا تعمیم آن کمک کند.

سؤال مهم دیگر در این رابطه آن است که "چه چیزهایی قادر به توصیف وضعیت ارادی هستند؟" این گونه به نظر می‌رسد که همه چیز می‌تواند مبین وضعیت ارادی باشد. Seel در رسالهٔ دکترای خود [۲٤] می‌گوید: "در نظر گرفتن سوئیچ برق به عنوان یک عامل کاملاً درست است. زیرا این عامل آرزوی انتقال برق را دارد و زمانی این کار را انجام می‌دهد که باور کند ما می‌خواهیم او جریان برق را انتقال دهد. روشن کردن سوئیچ راه ارتباط آرزوهای طرفین است. "

در حقیقت هر چه بیشتر در مورد سیستم بدانیم نیاز به تصور ارادی از آن کمتر می‌شود. سئوال دیگر این است که "سیستم ارادی بودن شرط لازم برای عامل بودن است. امّا آیا شرط کافی هم هست؟" Shardlow بر این عقیده است که "شاید عامل بیش از اعتقاد و آرزو بخواهد، که همین امر مورد بحث دانشمندان علوم ادراکی است."[۲۵]

۱-۳-معماری عامل

با توجه به آنچه در بخشهای گذشته ذکر شد، این سؤال پیش می آید که ساختار درونی عاملها چگونه است. در ذیل دو تعریف از معماری عاملها را بیان کرده و پس از آن مروری اجمالی بر اهم معماری عاملها خواهیم داشت.

Maes[۱۸] معماری عامل را بصورت زیر تعریف می کند:

"معماری عامل یک متدولوژی ویژه برای ساخت عاملها است که مشخص می سازد، چگونه عاملها می توانند به مجموعه ای از ماژولها شکسته شوند و چگونه این ماژولها با هم ارتباط داشته باشند. مجموعهٔ ماژولها و روابط بین آنها پاسخی به این سؤال هستند؛ که چگونه داده حسگرهاو وضعیت فعلی، کنش ها و حالت بعدی را مشخص می کنند. معماری شامل تکنیکها و الگوریتمهایی است که این متدولوژی را حمایت می کند."

Kaelbling[۱٦] نیز تعریف زیر را از معماری عامل دارد:

"مجموعه ای از ماژولهای سخت افزاری یا نرم افزاری است که بصورت جعبه هایی با پیکانهایی که نشان دهنده جریان داده ای و کنترلی بین ماژولهاست، به همدیگر متصل شده اند. دید انتزاعی تر از معماری آن است که متدولوژی ای عمومی است برای تقسیم ماژولار وظایف."

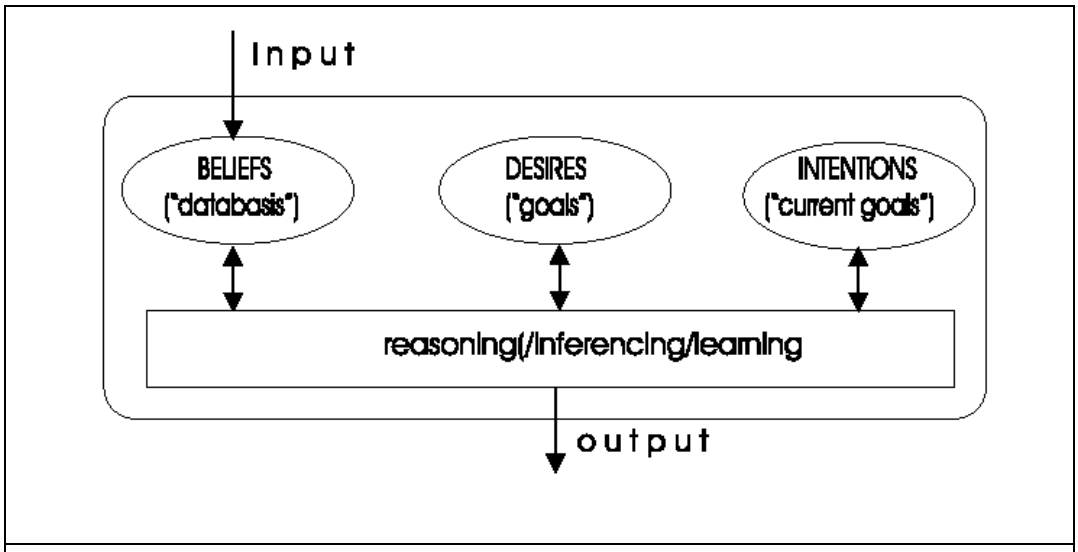
سه دستهٔ معروف از معماریها عبارتند از:

۱- عامل برنامه ریز
۲- عامل واکنشی
۳- عامل هیبرید

۱-۳-۱- معماری عامل برنامه ریز:

در مبحث برنامه ریزی AI Planning Community پرچمدار بسیاری از ابتکارات در برنامه ریزی است. آنها معتقدند برنامه ریزی جزء لاینفک هر سیستم هوشمند است. از انجا که این معماری در فرآیند برنامه ریزی خود از سه ساختار اعتقاد، آرزو و اراده به این دسته از معماری ها Belief-Desire-Intention(BDI) نیز می گویند.

شکل ۱-۱ ساختار درونی این دسته از معماری را نشن می دهد. که در آن اعتقادات (Beliefs) حقایق محیط، آرزوها (Desires) اهداف عامل، و اراده (Intentions) هدف فعلی عامل می باشند[۱٤].



شکل۱-۱-ساختار کلی معماری عاملهای برنامه ریز

ازمهمترین معماری های برنامه ریز می توان به عاملهای IRMA، *GRATE وHOMER اشاره کرد، که در ذیل شرح مختصر هر یک می آید.

IRMA :مخفف Intelligent Resource bounded Machine Architecture می باشد.

شکل ۱-۲ ساختار درونی این عامل را نشان می دهد که در آن چهار ساختمان داده سمبولیک وجود دارد: کتابخانه برنامه ، توصیف های صریح Belief،Desire و Intention.

علاوه بر آن، این معماری دارای فرآیندهای زیر است[۳۱]

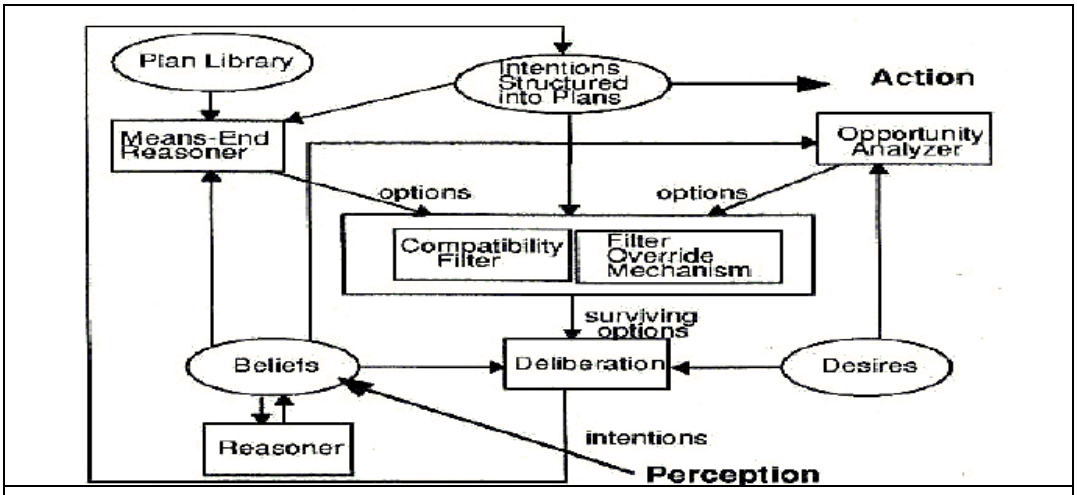
۱- Reasoner: برای استنتاج در مورد جهان

۲- Means-end analyzer: تشخیص برنامه هایی که برای دستیابی اراده در دسترس هستند.

۳- Opportunity analyzer: که محیط را در مقایسه با گزینه ها دیگر عامل بررسی می کند.

٤- Filtering process: تشخیص زیر مجموعه ا ی از پتانسیلهای عملیاتی که با اراده عامل همخوانی دارد.

٥- Deliberation Process: انتخاب بین گزینه ها را بر عهده دارد.



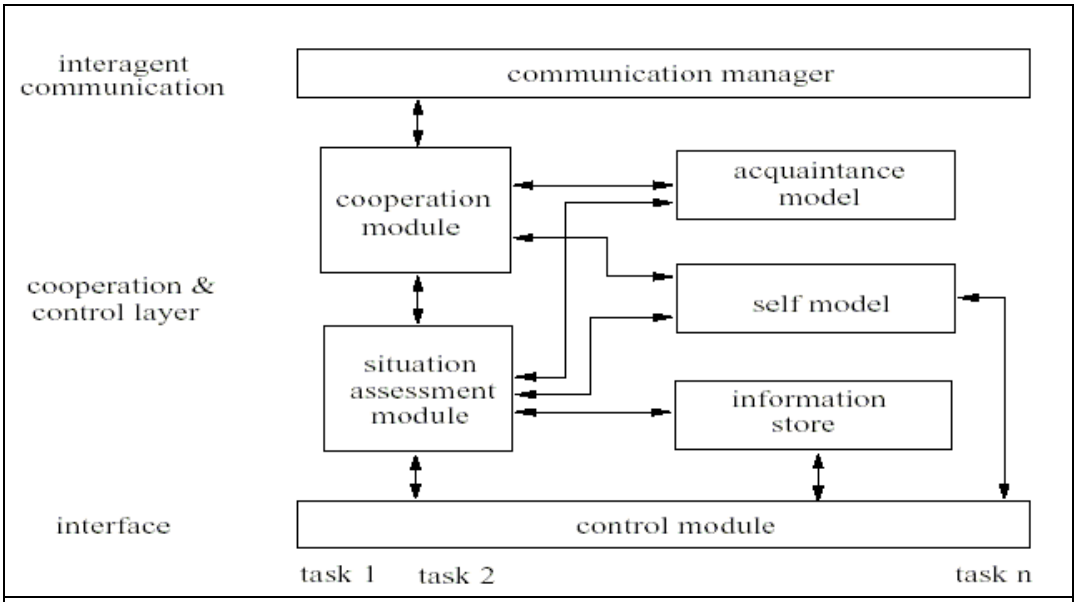
شکل ۱-۲- معماری IRMA

HOMER: در این عامل اعتقاد بر این بود که عامل خودگردان باید قابلیت زبانشناسی ، برنامه ریزی و اجرا داشته باشد[۳۱].

***GRATE**: یک معماری لایه بندی شده است که بوسیله صفاتی مانند اعتقادات، آرزوها ، اراده و ارادهٔ مشترک فعالیت می کند. شکل ۱-۳ ساختار این عامل را نشان می دهد[۳۱].

این معماری شامل دو لایه، لایهٔ سیستم سطح دامنه ولایهٔ هماهنگی و کنترل است. اولین لایه مربوط به دامنهٔ مسأله است و دومی بر فعالیتهای دامنهٔ عامل با سایرین که در اجتماع صورت می گیرد نظارت دارد.

لایهٔ همکاری شامل دو ماژول می باشد. ماژول کنترل که با سیستم سطح دامنه تعامل دارد و ماژول ارزیابی و همکاری که وظیفهٔ ارزیابی و پیاده سازی وظایف مشترک را بر عهده دارد.



شکل۱-۳- ساختار معماری *GRATE

۱-۳-۲- معماری عامل واکنشی:

به آن دسته معماریهایی اطلاق می شود که مدل سمبولیک از جهان ندارند و از استنتاجات پیچیدهٔ سمبولیک استفاده نمی کنند.

Brooks در مورد معمار یهای غیر سمبولیک سه نکته را بیان می دارد[۲]:

۱- رفتار هوشمند می تواند بدون استفاده از توصیفهای صریح سمبولیک تولید شود.

۲- رفتار هوشمند می تواند بدون استفاده ازاستنتاجات سمبولیک تولید گردد.

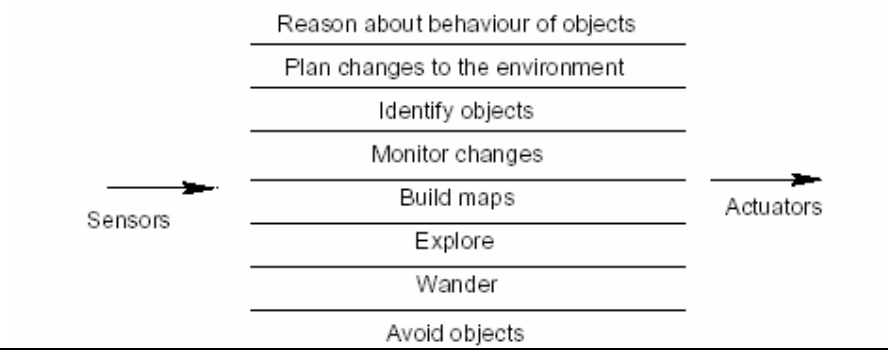
۳- هوشمندی خاصیت اصلی سیستمهای پیچیده است.

در این راستا Brooks دو ایده را مطرح می سازد.:

۱- تجسم و واقعیت: هوشمندی حقیقی در جهان واقع شده نه درسیستمهایی مانند اثبات کنندهٔ قضیه و یا سیستمهای خبره.

۲- هوشمندی و تجسم: رفتار هوشمندانه بصورت تعامل عامل با محیطش ساطع می شود؛ همچنین هوشمندی در ذات است و چیزی اضافی و مجزا نیست

معماری Brooks(Subsumption Architecture): در این معماری کارها بصورت سلسله مراتبی انجام می شود. همانطور که در شکل ۱-٤ دیده می شود در سلسله مراتب پایین تر کارهای ابتدایی تر انجام می شود[۲].(مثلا تشخیص مانع)



شکل ۱-٤- معماری Subsumption

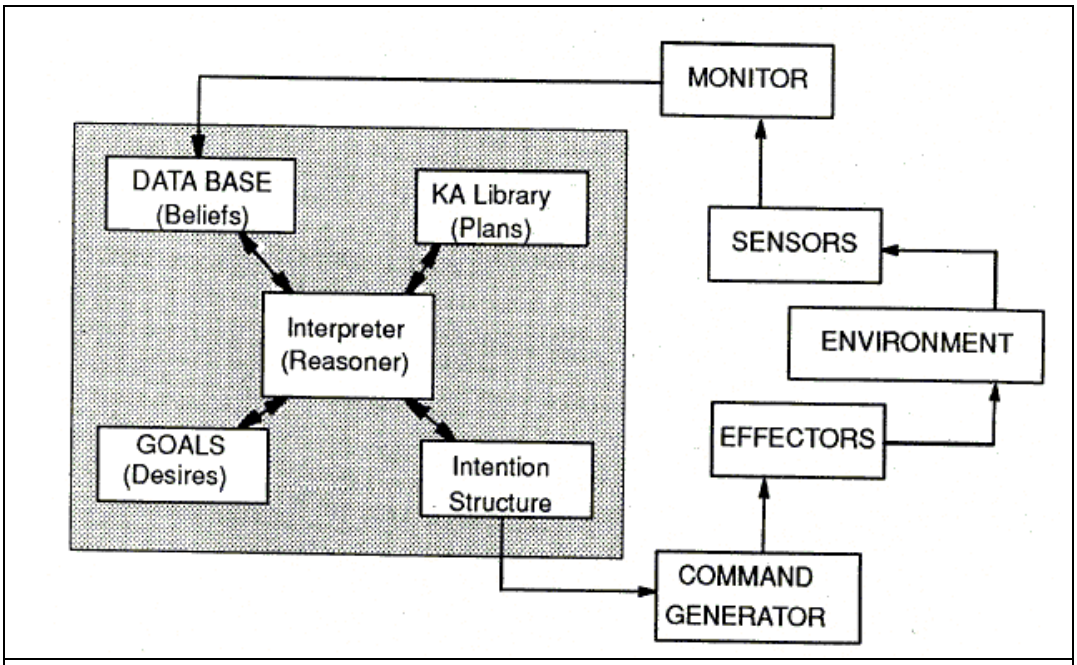
PENGI: سازندگان آن بر آن عقیده اند که اکثر کارهای روزانهٔ ما روتین است. بسیاری از تصمیم ها بصورت روتین انجام می شوند. بنابراین آنها می توان کد کرد و هر از چندی به روز رساند[۳۱].

Situated Automata: دارای رویکردی پیچیده است که شامل درک و کنش می باشد. از دو برنامه برای انجام این دو وظیفه استفاده می کند. این دو برنامه عبارتند از RULER و GAPPS که پردازشهای سمبولیک را به عهده دارند[۳۱].

۱-۳-۳- معماری عامل هیبرید:

بعضاً معتقدند که هیچ یک از دو رویکرد فوق الذکر مناسب نیستند بلکه تلفیقی از آنها مؤثر می باشد.بدین منظور آنان سعی در تلفیق آنها دارند. اهم این معماریها عبارتند از: PRS،Touring Machine،COSY

PRS: مخففProcedural Reasoning System است. در این معماری ساختار اعتقاد،آرزو ، اراده و کتابخانهٔ برنامه صریحاً بصورت سمبولیک بیان شده است. که در آن اعتقاد همان حقایق جهان پیرامون یا وضعیت درونی سیستم (بصورت منطق مرتبه اول) می باشد.آرزو رفتارهای سیستم می باشد. کتابخانهٔ برنامه مجموعهٔ برنامه های ناکامل است که KA(Knowledge Area) خوانده می شود و هریک از آنها شامل شرط واکشی هستند. اراده مجموعهٔ KAهای فعال است. شکل ۱-۵ این ساختار درونی این معماری را نشان می دهد[۳۱].



شکل ۱-۵- معماری PRS

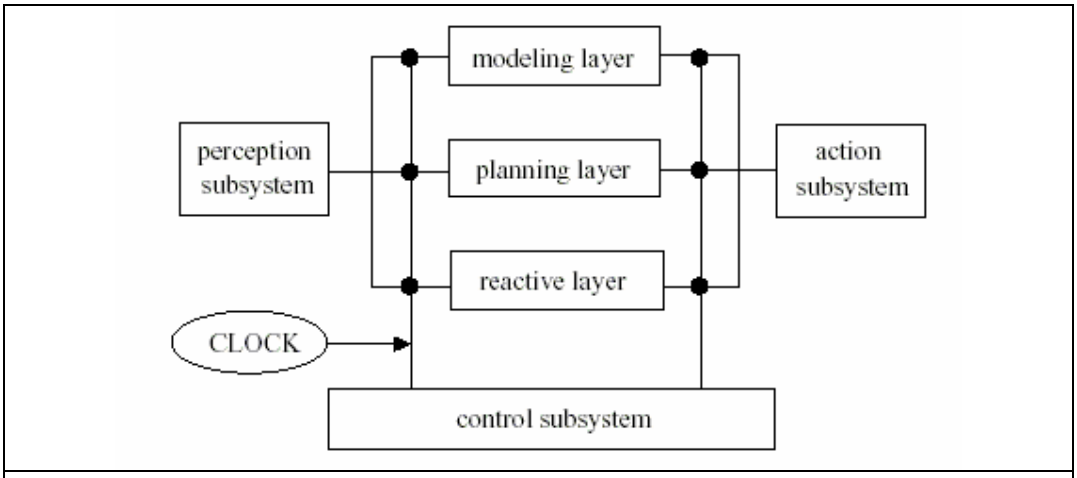
Touring Machines: همانطور که در شکل۱-٦ آمده استاین معماری شامل سه لایه است[۳۱]:.

۱- لایهٔ واکنش: مجموعه ای از قوانین وضعیت-کنش است و زمانیکه واکنش باید به سرعت انجام شود از آن استفاده می شود.

۲- لایهٔ برنامه ریزی: برنامه ریزی می کند و کنش های مورد نیاز برای رسیدن به هدف مشخص می شود.

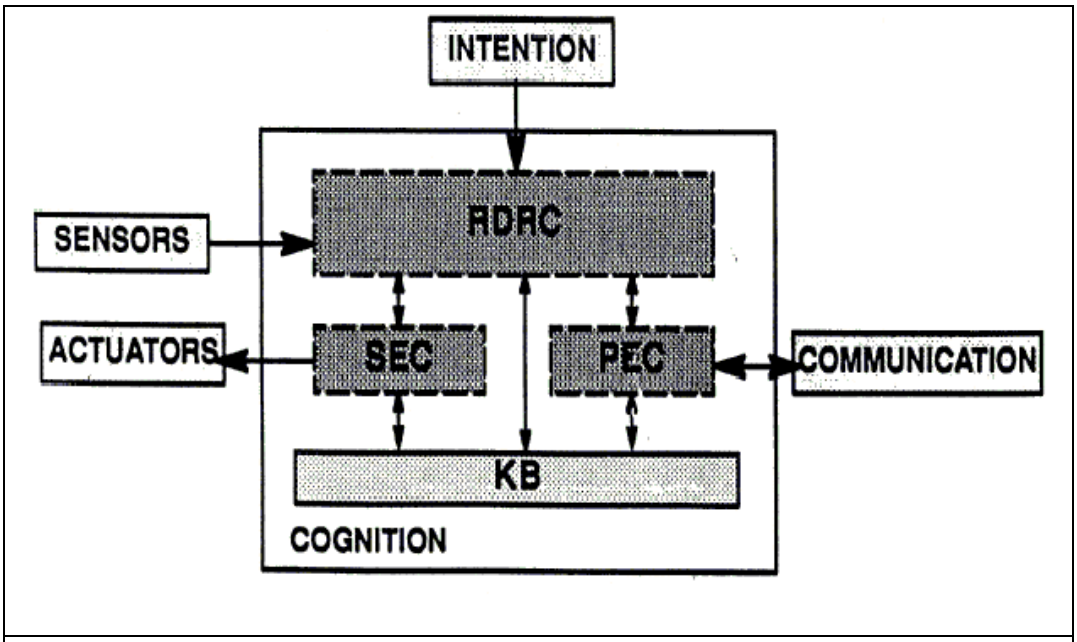
۳- لایهٔ مدلسازی: نمایش سمبولیک حالات روانشناختی دیگر اجزای موجود در محیط عامل ها را داراست. و مسؤول حل کردن تداخلها می باشد.

لایه ها با ارسال پیام ارتباط برقرار می کنند. همهٔ لایه ها در قالب کنترلی قرار گرفته اند و هدف این قالب حل کردن تداخل بین لایه های مختلف است.



شکل ۱-٦- معماری Touring Machines

COSY: همانگونه که در شکل۱-۷ نشان داده شده است. این معماری شامل ۵ جزء حسگر،کنشگر، ارتباطات، ادراک و اراده است.ماژول ادراک وظیفهٔ میانجیگری بین اراده ها و عقاید عامل و همچنین گزینش کنش مطلوب را به عهده دارد.



شکل۱-۷- معماری COSY

۱-٤- زبان عامل

منظور از زبان عامل سیستمی است که به کاربر اجازه دهد سیستم‌های سخت‌افزاری یا نرم‌افزاری را با مفاهیمی که نظریه پردازان عرضه کرده‌اند، برنامه‌ریزی کند. بنابراین انتظار می‌رود این زبانها حداقل ساختارهای مربوط به عاملها مانند اعتقاد، اراده ، آرزوها و... را داشته باشند[۳۱].

۱-٤-۱- زبانهای مبتنی بر اشیای همروند

ریشهٔ اکثر زبانهای عاملها زبانهای مبتنی بر اشیای همروند هستند. آنها متشکل از اشیا خودگردان با حالت درونی مخصوص به خود هستند. این حالات درونی از بیرون قابل دسترسی نیست. این اشیاء با ارسال پیام با هم ارتباط برقرار می‌کنند. اولین نمونه از این زبانها مدل هنرپیشهٔ Hewitt بود. ABCL [۳۲] مثال دیگری از این زبانها است.

۱-٤-۲- برنامه‌سازی عامل‌گرا

Shoham روش جدیدی را برای برنامه‌نویسی پیشنهاد کرده است[۲٦]. ایدهٔ اصلی او برنامه‌نویسی مبتنی بر عامل بود. این نوع برنامه‌نویسی مستقیماً، برنامه‌نویسی عاملها با خواص ارادی و ذهنی آنها است. Shoham متعقد است که یک سیستم برنامه‌سازی عامل‌گرا باید شامل سه عنصر باشد.

۱- سیستمی منطقی برای تعریف وضعیت ارادی عامل
۲- یک زبان برنامه‌نویسی تفسیری برای برنامه‌ریزی عاملها

۳- پروسهٔ "عاملی کردن[1]" برای ترجمهٔ برنامه‌های عامل‌گرا را به سیستم‌های سطح پایین قابل اجرا.
اولین تلاش Shoham در رسیدن به چنین سیستمی AGENT۰ بود. Thomas با PLACA[۳۰] و Fisher با Concurrent METAM[۷]،از پیروان نظریهٔ Shoham بودند.

۱-۵- عامل در سایر محیطها:

متون علمی عاملها به دسته‌های مختلفی از مباحث مانند داشتن اطلاعات، تصمیم‌گیری و کنش پرداخته‌اند. این مباحث را می‌توان به صورت زیر نیز دسته‌بندی کرد.

۱- مباحث ادراکی[2]: به اطلاعات و دانش عاملها می‌پردازد.
۲- مباحث تأثیرگذاری[3]: بر انتخاب و تصمیم‌گیری دلالت دارد.
۳- مباحث کردار انگیزانه[4]: که به کنش و نحوهٔ انجام کنش می‌پردازد.

از ایده های موجود در علوم دیگر به منظور ارتقای مباحث فوق کمک گرفته شده است. آنچه در ذیل می‌آید برخی از این علوم است.

منطق

- توصیف فرمال
- منطقهای ویژه مانند منطق پویا، منطق زمانی و...
- اثبات فرضیه
- نظریهٔ معنا و مدل

ریاضیات

- نظریه های اطلاعات و احتمال
- نظریه های ساختار: توپولوژی، گراف، لاتیس
- توصیف تغییرات زمانی، معادلات دیفرانسیل
- نظریه های تخمین

علوم کامپیوتر

- محاسبه در سیستم توزیع شده
- آنالیز ساختارهای پردازه‌ها و زمان
- آنالیز ساختار و ساختمان داده‌ها

---

[1] Agentification

[2] Cognitive

[3] Affective

[4] Conative

فیزیک

- مفاهیم کنش در فیزیک
- مفاهیم محدودیت (و درجهٔ آزادی)، کمترین هزینه و اصول بهینگی، طبقه‌بندی متغیرها
- واکنشهای سیستم‌های غیرخطی
- سیستم‌های توزیع شده متشکل از تعداد زیاد عاملهای ساده مانند حرکت چند ذره‌ای، گازها، مایعها، مغناطیس و...

فلسفه:

- مفاهیم عاملیت پیچیده در فلسفهٔ مغز
- فلسفهٔ کنش، صفات، اراده، خود، آرزو و...

زیست‌شناسی:

- جامعهٔ حشرات
- تکامل

روانشناسی:

- اشتیاق
- انگیزه
- ادراک
- ساختارها و پروسه‌های اجتماعی

هوش مصنوعی:

- نمایش دانش و استنتاج در ماشین‌ها
- تئوری‌های حل مسأله، جستجو
- برنامه‌ریزی
- تشخیص الگو

اقتصاد:

- تئوری تعادل در اقتصاد
- رفتار گروه

۲-۱- مقدمه

یادگیری مانند هوش چنان محدودهٔ وسیعی را شامل می شود که تعریف دقیق آن ممکن نیست.معادل یادگیری در لغـتنامه ها عباراتی مانند "کسب دانش یا ادراک یا کسب مهارت به کمک مطالعه،رهنمود ،تجربه" آمده است.

۲-۲- تعریف یادگیری ماشین

در مورد ماشینها بطور کلی میتوان گفت، ماشین زمانی یاد می گیرد که ساختار، برنامه یا داده هایش (بر مبنای ورودیهای آن یا در پاسخ به اطلاعاتی از خارج سیستم) بگونه ای تغییر کند، که کارآیی اش افزایش یابد. برخی از این تغییرات ، مانند اضافه کردن یک رکورد به پایگاه داده ، در این قلمرو نمی گنجد. اما ،برای مثال ، زمانیکه کارآیی ماشینِ تشخیص صدا پس ازشنیدن نمونه هایی از صدای شخص افزایش می یابد، کاملاً یادگیری ماشین محسوس است. در ذیل به چند تعریف دیگر از یادگیری اشاره می شود:

:Herbert Simon

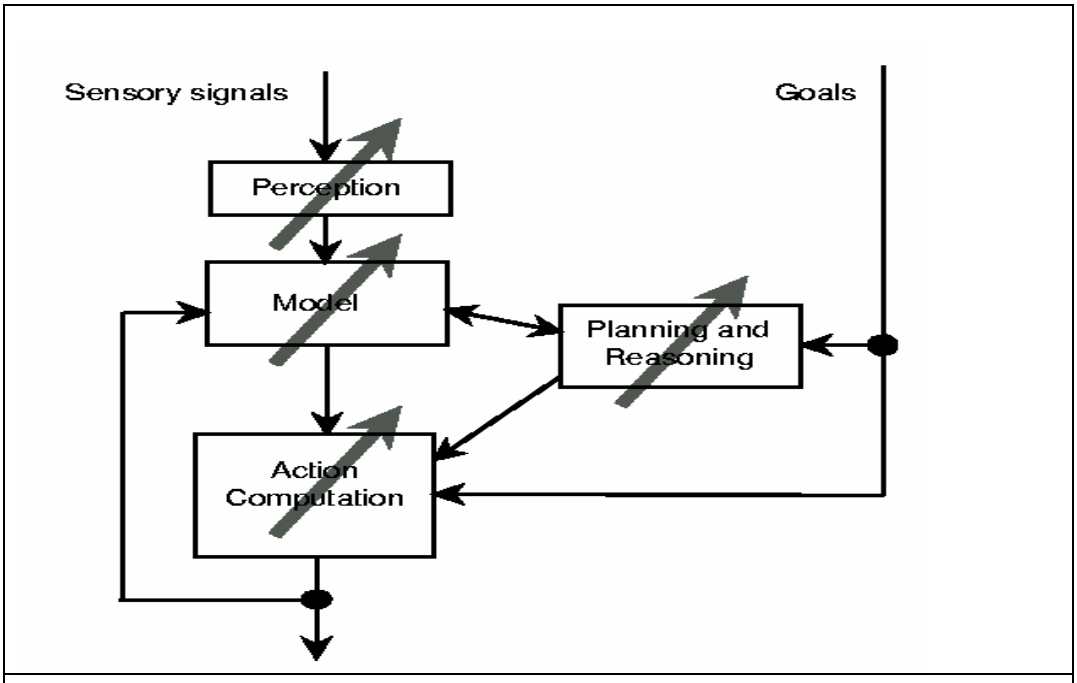
" یادگیری برتابندهٔتغییراتی درسیستم است که سیستم را قادر می سازد تا وظایف مشابه را بصورت کارآتری انجام دهد." [۲۷]

:Ryzard Michalski

" یادگیری عبارتست از ساخت یا تغییرنمایشها از آنچه که تجربه شده است."[۲۹]

:Marvin Minsky

"یادگیری یعنی انجام تغییرات مفید در ذهن"[۲۹]



شکل۲-۱- ساختار سیستم هوشمند

برای درک بهتر ارتباط یادگیری و تغییر در سیستم ، ساختار سیستم هوشمند که در شکل ۲-۱ را در نظر بگیرید. تغییر در هر یک از اجزای نمایش داده شده را می توان یادگیری دانست.

۲-۳- ضرورت یادگیری ماشین

حال شاید کسی بپرسد: "چرا ماشین باید بیاموزد؟ چرا ماشینهایی طراحی نشوند که از اول کار مورد نظر را بخوبی انجام دهند؟" دلایل متعددی برلزوم یادگیری ماشین وجود دارد.مثلاً یکی از دلایل می تواند این باشد که با یادگیری ماشین می توان به چگونگی یادگیری انسان پی برد. اما علاوه بر آن دلایل مهندسی محکمی نیز وجود دارد، که برخی از آنها به شرح زیر است:

۱- برخی از کارها تنها با مثال قابل بیان است. مثلاً ممکن است، قادر به مشخص کردن زوجهای (ورودی،خروجی) باشیم، اما رابطهٔ دقیقی بین آنها نداشته باشیم. بنابراین انتظار داریم که ماشینها قادر به تنظیم ساختار داخلی خود بگونه ای باشند که با دریافت یکی از زوجهای (ورودی ،خروجی) بتواند رابطهٔ ضمنی بین ورودی و خروجی را تخمین بزند.

۲- ممکن است بین انباشته ای از داده ها رابطه های مهمی پنهان باشند. می توان از روشهای یادگیری ماشین برای یافتن آنها استفاده کرد.

۳- غالباً طراحان ماشینهایی درست می کنند که کارآیی مورد نظر در محیط استفاده را ندارند. در حقیقت، ویژگیهایی از محیط عملیاتی در زمان طراحی بطور کامل توسط طراح درک نشده است. روشهای یادگیری ماشین می توانند به منظور "بهبود در حین کار" طراحی ماشینهای موجود استفاده شوند.

٤- ممکن است دانش موجود در زمینه ای آنقدر زیاد باشد که قابل کد کردن توسط انسان نباشد. ماشینهایی که این دانش را یاد می گیرند ممکن است بیش از آنچه انسان کد میکند ، کسب کنند.

٥- محیطها در طول زمان تغییر می کنند. ماشینهایی که بتوانند خود را با تغییر محیط سازگار سازند،نیاز به طراحی مجدد ندارند.

٦- همواره دانشهای جدیدی توسط بشر در حال کشف است. پدیده های جدیدی در جهان در حال رخ دادن است. طراحی مجدد سیستمهای هوشمند برای تطبیق با این دانشها ، غیر عملی می نماید. اما روشهای یادگیری ماشین می تواند در مورد یادگیری دانشهای جدید کارساز باشد.

۲-٤- سر چشمه های الگوریتمهای یادگیری ماشین:

فعالیتهای مرتبط به یادگیری ماشین از سرچشمه های مختلفی و از محیطهای متنوعی آغاز شده اند و این فعالیتها درحال همگرا و یکپارچه شدن تحت نام "یادگیری ماشین" هستند. این زمینه های مختلف فعالیت به همراه خود اصطلاحات و تکنیکهای مربوط به خود را به یادگیری ماشین وارد کرده اند. آنچه در پی می آید بررسی این سرچشمه ها بطور گذرا و کلی است.

۱- **آمار (Statistics)**: یک مسالهٔ مهم در آمار حدس در مورد تابع توزیع احتمال برای تابع احتمالاتی نامشخص بوده است. ومسالهٔ دیگر مرتبط با این مساله تخمین مقدار تابع در یک نقطه با استفاده از مجموعه ای از مقادیر مشخص تابع است. روشهای مختلف آماری برای حل اینگونه مسائل وجود دارد، که این قواعد و روشها را می توان نمونه هایی از یادگیری دانست.

۲- **مدلسازی از مغز(Brain Models)**. عناصر غیر خطی با ورودیهای وزندار به عنوان مدلهای ساده ای از نورونهای طبیعی معرفی شده اند. شبکه هایی از این عناصر را شبکه های عصبی گویند. مدلسازان مغز تلاش می کنند که فرآیند یادگیری این شبکه ها را با نحوهٔ یادگیری در مغز نزدیک سازند. از

کارهایی که از همین موضوع الهام گرفته اند می توان کانکشنیزم[1]، محاسبات شبه مغز[2] یا پردازش شبه سمبولیک[3] را نام برد.

۳- **تئوری کنترل وفقی (Adaptive Control Theory).** نظریه پردازهای کنترل همواره با مسالهٔ کنترل یک فرآیند با پارامترهای نامشخص مواجهند. این پارامترها در حین عملیات تغییر می کنند و باید مقدار آنها تخمین و تغییرات آنها کنترل شوند.نمونه ای از این مسالهٔ کنترل روبوت بر حسب ورودیهای آن است.

٤- **مدلهای روانشناسی(Psychological Models).** نحوهٔ یادگیری انسانها در موارد مختلف مورد بررسی روانشناسان بوده است. همین امر باعث ایجاد روشهایی از یادگیری بوده است. مثال اولیهٔ این روشها شبکهٔ EPAM است که زوج کلمات را ذخیره کرده و در زمان بازیابی با ارائهٔ یکی از آنها به شبکه ، دیگری بازیابی می شود. ادامهٔ کارها در این زمینه منجر به درختهای تصمیم گیری و شبکه های معنایی شد. فعالیتهایی که در زمینهٔ یادگیری پاداشی می شود را می توان بر گرفته از یادگیری هدفدار حیوانات دانست.

٥- **هوش مصنوعی(Artificial Intelligence).** هوش مصنوعی از ابتدا با هدف یادگیری ماشین آغاز شد.ساموئل اولین برنامهٔ مهم را نوشت که پارامترهای تابع ارزشیابی موقعیتها در صفحهٔ بازی چِکرز(Checkers) را یاد می گرفت. پژوهشگران متوجه نقش شباهت در یاد گیری ، همچنین چگونگی تصمیم گیری برای آینده با استفاده از نتایج قبلی شده اند. اکنون، عمده فعالیتها بر روی استخراج قوانین از سیستمهای خبره و برنامه نویسی منطقی استقرایی[4] صورت می گیرد.

٦- **مدلهای مبتنی بر نظریهٔ تکامل(Evolutionary Models):**در طبیعت ، موجودات نه تنها یاد می گیرند که کارهای خود را بهتر انجام دهند، بلکه سعی می کنند با تکامل محیط مرتبهٔخود را حفظ کند. این تفاوت بین یادگیری و تکامل در سیستمهای کامپیوتری کمرنگ می شود و به تکنیکهایی که از جنبه های معینی از تکامل را مد نظر دارندو به منظور بهتر کردن برنامه های کامپیوتری بکارمی روند، گویند. الگوریتمهای ژنتیک برجسته ترین این روشهاست.

۲-۵- دسته بندی الگوریتمهای یادگیری ماشین

از زوایای مختلف می توان الگوریتمهای یادگیری را طبقه بندی کرد. بدین خاطر دسته بندیهای مختلفی عرضه شده است ، که در ذیل شرح برخی از آنها آمده است.

۲-۵-۱- یادگیری Blackbox و دانش گرا: در این تقسیم بندی الگوریتمهای یادگیری ماشین به دوگروه تقسیم شده اند(۱) روشهای Blackbox و(۲) روشهای دانش گرا

متدهای Blackbox نمایش مفاهیم را به گونه ای توسعه می دهند که توصیف داخلی سیستم و تفسیر آن برای کاربر میسر نیست . در این نوع روشها عموماً از محاسبه ضرایب ، فواصل یا اوزان استفاده می شود، مانند شبکه

---

[1] Connectionism

[2] Brain Style

[3] SubSymbolic

[4] Inductive Logic Programming

های عصبی و روشهای آماری . در عوض ، روشهای دانش گرا بر ساخت ساختارهای سمبولیک برای نمایش واضح مفهوم اصرار دارند ، مانند درخت تصمیم.

تفاوت سیستمهای دانش گرا و Blackbox را می توان با استفاده از سه مقیاس – ضعیف ، قوی و بسیار قوی – بیان شده است . دسته بندی در این مقیاسها بر اساس وضوح توصیف مفهوم است .

۱- ضعیف : سیستم از نمونه های داده برای تولید داده های جدید استفاده می کند .

۲- قوی : همان مورد قبل است به علاوه اینکه توصیف مفاهیم بوضوح صورت می گیرد .

۳- بسیار قوی : همان مورد قبل است به علاوه اینکه سیستم با ساختاری کاراً بطور سمبولیک عمل می کند .

شبکه های عصبی و روشهای آماری در مقیاس ضعیف گنجیده، روشهای سمبولیک هوش مصنوعی در مقیاس قوی گنجیده ، و در مقیاس سوم کاربر نه تنها باید توصیف منتجه را بفهمد بلکه باید از آن بدون کمک کامپیوتر استفاده کند و یا به عبارتی کاربر تمام محاسبات مورد نیاز را در ذهن خود انجام دهد.

۲-۵-۲-دسته بندی بر اساس نظارت

این دسته بندی بر اساس تفاوت استفاده الگوریتم از اطلاعات کلاسهای الگو انجام می شود. که شامل سه دستهٔ (۱) با مربی،(۲) بدون مربی و (۳) پاداشی می باشد. در روشهای یادگیری "بامربی[۱]" فرض بر آن است که معلم یا ناظری وجود دارد که مثالهای آموزشی را به کلاسهایی طبقه بندی می کند . در صورتیکه روشهای یادگیری "بی مربی[۲]" این فرض را ندارند . بنابراین یادگیری بی مربی باید طبقه بندی اطلاعات الگوها را به عنوان جزئی از فرآیند یادگیری انجام دهد . روشهای یادگیری "پاداشی[۳]" دسته ای بین دو دستهٔ قبل است. در این گونه روشها مربی تنها با دادن سیگنالهای پاداشی، به روند یادگیری کمک می کند. از این رو بعضی آنها را نوعی یادگیری بی مربی می دانند. در کل وظیفه یادگیری بی مربی انتراعی تر است ، یادگیرنده باید نظم محیط را درک کرده و بر حسب آن یک فرضیه استخراج کند .

الگوریتمهای یادگیری با مربی از اطلاعات دسته بندی هر یک از نمونه ها استفاده می کنند . این اطلاعات ، این امکان را فراهم می سازد تا الگوریتم دسته بندی نادرست الگوها را تشخیص داده و به عنوان بازخوری از آن استفاده کند . اطلاعات مربوط به خطا، به روند یادگیری با سیاست تشویق در مقابل طبقه بندی درست، و تنبیه در مقابل طبقه بندی نادرست، کمک می کند . همچنین در حذف فرضیه های غلط نیز مؤثر است .

الگوریتمهای یادگیری بی مربی از الگوهای طبفه بندی نشده استفاده می کند. اغلب الگوریتمهای بی مربی پیچیدگی محاسباتی و دقت کمتری نسبت به الگوریتمهای با مربی دارند . همین امر باعث شده است یادگیری بی مربی در بسیاری از محیطهای پرسرعت و بی درنگ که زمان کافی برای استفاده از تکنیکهای بامربی نیست ، استفاده شود .

---

[۱] Supervised

[۲] Unsupervised

[۳] **Reinforcement**

۳-۱- مقدمه

از آنچه در بخش ۱-۲ گفته شد می‌توان دریافت که عامل سیستمی است که در جهت نیل به مجموعه ای از اهداف در محیطی پویا و پیچیده فعالیت می کند. عامل در محیط قرار دارد و می‌تواند با حسگرهای خود، محیط را ادراک کند و بر اساس آن با استفاده از کنشگرهای خود بر محیط تأثیر گذارد. اهداف عامل می‌توانند قالبهای مختلفی داشته باشد. آنها می‌توانند هدف نهایی یا حالتی مشخص برای رسیدن، باشند.

۳-۲- تعریف عامل یادگیر

عامل را یادگیر گویند اگر قادر باشد در طول زمان ، عملکردش را بهبود بخشد. به عبارت دیگر، اگر عملکرد عامل در رسیدن به اهدافش به واسطهٔ تجربه‌اش بهتر شود. باید توجه داشت که یادگیری عامل درجاتی دارد از یادگیری تغییرات کوچک کوتاه مدت گرفته تا تغییرات اساسی بلندمدت در محیط.

۳-۳- معماری کلی یک عامل یادگیر:

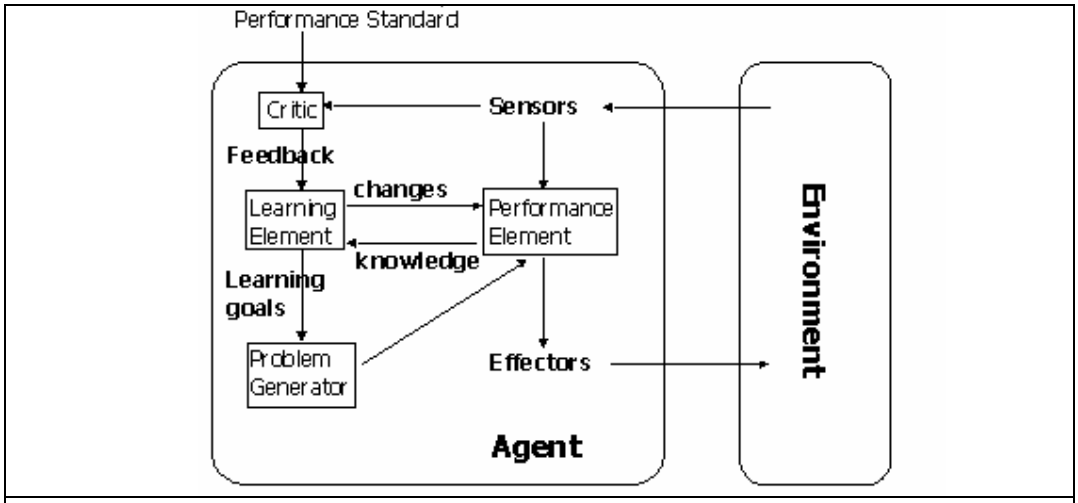
شکل ۳-۱ معماری عمومی یک عامل یادگیر را نشان می‌دهد[۲۳] که در آن:

۱- Learning Element: تغییرات در سیستم را بر اساس چگونگی عملکرد آن اعمال می‌کند.

۲- Performance Element: عامل منهای سیستم یادگیری آن است.

۳- Critic: با مقایسه کارآیی سیستم با استاندارد به Learning Element می‌گوید که چه کند.

٤- Problem Generator: کنشها یا مسأله‌های جدید که به سیر یادگیری عامل کمک می‌کند را توصیه می‌کند.



شکل ۳-۱- معماری کلی یک عامل یادگیر

۳-٤- یادگیری عامل باید چه خواصی داشته باشد؟

Maes خواص یادگیری در عاملها را به صورت زیر مطرح ساخته و معتقد است که هر نوع عامل و با هر نوع هدف، مدل یادگیری عامل باید شروط زیر را بر آورده سازد[۱۷].

۱- یادگیری باید افزایشی باشد. عامل باید در هر تجربه بیاموزد. یادگیری و انجام عمل نمی‌توانند در دو فاز جداگانه انجام شوند.
۲- یادگیری باید در جهت یادگیری دانش مرتبط با هدف عامل باشد. در محیطهای پیچیدهٔ واقعی نمی‌توان از عامل انتظار داشت که همه چیز را یاد بگیرد.
۳- مدل یادگیری باید قادر به تحمل نویز، محیط احتمالاتی، حسگرهای نامطمئن و... باشد.
٤- یادگیری باید بدون مربی باشد. عامل باید خود به تنهایی قادر به آموختن باشد.
٥- ترجیحاً، مدل یادگیری باید قادر باشد دانش اولیه‌ای را از سوی کاربر بپذیرد.

۳-۵- روشهای متداول یادگیری

در بخش قبل برخی از معماری‌های عامل یادگیر مورد بررسی قرار گرفت. یادگیری از تجربه، شرط لازم برای عاملی است که باید در مدت زمان طولانی رفتاری خودگردان و صحیح داشته باشد. زیرا، اولاً برنامه‌نویسی برای عامل بسیار مشکل است. ثانیاً محیط ممکن است دائماً تغییر کند که در صورت عدم وجود یادگیری نیاز به برنامه‌نویسی مکرر عامل است.

مسأله یادگیری عامل را می‌توان بصورت زیر بیان کرد[۱۷].

یک عامل با دارا بودن (۱) مجموعه‌ای از کنشها یا ماژولهای متخصص، (۲) دادهٔ حسگر و (۳) چند هدف، چگونه می‌تواند با توجه به تجربه‌اش، رفتارش را بهبود بخشد؟ یا به عبارت دیگر چگونه می‌تواند از بازخوردی که از محیط می‌گیرد در جهت بهبود رفتارش استفاده کند؟

منظور از بهبود رفتار عامل آن است که در نیل به اهداف یا خواسته‌هایش موفق‌تر باشد. بسته به طبیعت هدف عامل، این توفیق معانی مختلفی می‌دهد. مثلاً برای هدف نهایی بدین معناست که زمان متوسط نیل به آن هدف یا تعداد کنشهای مورد نظر برای رسیدن به آن در طول زمان کاهش یابد. در مورد هدفهایی از نوع بیشینه کردن پاداش، به معنای افزایش متوسط پاداشهای دریافتی در بازهٔ مشخص زمانی می‌باشد.

در طراحی هر مدل یادگیری عامل سه مسأله مطرح می شوند. که عبارتند از:

۱- مکانیزم انتخاب کنش چیست؟
۲- روش یادگیری چیست؟
۳- چگونه عامل دانش جدید استخراج کرده و به کار گیرد؟

در حل مسائل فوق، مسأله‌های دیگری نیز وجود دارد که باید حل شود. مثلاً هر عامل یادگیر با مسأله "انتساب تأثیرگذاری" مواجه است. یعنی کنشهای انجام شده، هر یک چقدر در حصول نتیجه سهیم بوده‌اند؟

بسته به محیط و عامل، نقش یادگیری متفاوت خواهد بود. در محیطی که قابل پیش‌بینی است و سرعت تغییرات بسیار کندتر از طول عمر عامل است، نیاز کمی به یادگیری وجود دارد. Todd و Wilson [۲۹] از جمله کسانی هستند که تحقیقات زیادی روی رابطهٔ محیط با عامل انجام داده‌اند.

روشهای متداول یادگیری عبارتند از:

۱- یادگیری پاداشی
۲- طبقه بندها
۳- سازنده مدل

۳-۵-۱- یادگیری پاداشی

ایدهٔ یادگیری پاداشی آن است که عامل با داشتن (۱) مجموعه‌ای از کنشهای ممکن (۲) مجموعه‌ای از وضعیتهای ممکن و (۳) سیگنال عددی پاداش ، نگاشتی از وضعیتها به کنشها را فرا می‌گیرد. به گونه‌ای که پاداش دریافتی عامل در طول زمان بیشینه شود.

یادگیری Q، استراتژی معروف یادگیری پاداشی ماشین است. در روش Q عامل سعی دارد برای تمام زوجهای (وضعیت – کنش) مقدار پاداش را یاد گیرد. بطور دقیقتر، عامل ماتریسی دوبعدی را یاد می‌گیرد که به ازای هر ترکیب ممکن وضعیت و کنش مقدار پاداش آن ذخیره شده است. در آغاز فرآیند تمام عناصر ماتریس با مقدار اولیه‌ای آغازانده می‌شوند. هدف سیستم به روز رساندن عناصر این ماتریس به گونه‌ای است که، این مقادیر به ماکزیمم مقدار خود همگرا شوند. این روش سه مسأله طراحی مذکور در بخش قبل را به صورت زیر حل کرده است.

۱- **مکانیزم انتخاب کنش:** در هر لحظه عامل در وضعیتی است و با توجه به آن وضعیت کنشی را برمی‌گزیند که بیشترین مقدار پاداش را دارد.

۲- **روش یادگیری:** هر وقت عامل کنشی را انجام دهد که منتهی به پاداش‌شود. پس از گرفتن پاداش عامل مقدار (وضعیت – کنش) را بروز می‌رساند.

۳- **استراتژی استخراج:** عامل در تعداد خاصی از وضعیتها، کنشی که پاداش را بیشینه می‌کند، برنمی‌گزینند؛ در عوض کنشی دیگر را بطور تصادفی انتخاب و اجرا می کند. این عمل باعث پیدا شدن راه حلهای جدید می‌شود.

یکی از جذابیتهای یادگیری پاداشی پایهٔ نظری آن است. ثابت شده است که در شرایط ویژه‌ای عامل به سمت سیاست انتخاب کنش بهینه همگرا می‌شود. متأسفانه این شرایط در وضعیتهای حقیقی و پیچیده دست یافتنی نیست. معایب یادگیری پاداشی عبارتند از: (۱) برای اهداف متغیر قابل استفاده نیست. (۲) اگر هدف تغییر کند همه چیز باید دوباره آموخته شود. (۳) در کاربردهای واقعی اندازهٔ فضای حالات بسیار بزرگ است. بطوریکه یادگیری زمان بسیار زیادی را می‌گیرد و عملی نیست. (٤) تنها در زمانی سیستم یادگیری را انجام می‌دهدکه پاداش گرفته شود. (۵) مدل یادگیری فرض می‌کند عامل در تمام زمانها می‌داند که در چه وضعیتی است (با داشتن حسگرهای نامطمئن و یا وضعیتهای مخفی برآورده ساختن این شرط مشکل است). (٦) ارائهٔ دانش اولیه به چنین ساختاری مشکل است و (۷) این روش زمانیکه چندین کنش به موازات انجام می‌شود قادر به یادگیری نیست.

۳-۵-۲- روشهای طبقه‌بندی

روش دوم یادگیری عاملها، مبتنی بر طبقه‌بندی است. این روش را می‌توان نمونه‌ای تغییر یافته از یادگیری پاداشی دانست. به این صورت که سیستم باز سعی در بیشینه کردن پاداش دارد. ایدهٔ این نوع سیستم‌ها آن است که عامل دارای مجموعه‌ای از قوانین به نام "طبقه‌بند" است و در مورد هر یک از آنها داده‌هایی در مورد کارآیی دارد. برای هر قانون حداقل یک مقدار که نشان دهندهٔ قدرت آن است وجود دارد.

سه مسأله طراحی در این سیستم‌ها به طرز زیر حل شده است.

۱- **مکانیزم انتخاب کنش:** با داشتن یک وضعیت که شامل تعدادی حالات بیرونی (داده حسگرها) و یک حالت درونی، پیش شرط تعدادی از طبقه‌بندها ارضا می‌شود. از میان تمام این طبقه‌بندها، عامل یکی یا بیشتر از یکی از طبقه‌بندها را به نسبت قدرتشان برگزیده و کنشهای مورد نظر آن قوانین را انجام می‌دهد.

۲- **روش یادگیری:** هر زمان که چند طبقه‌بند اجرا می‌شود، مقداری از قدرتشان را به طبقه‌بندهایی که در بازهٔ زمانی قبل اجرا شده‌اند می‌دهند. این الگوریتم را Bucket- brigade گویند که برای حل مسأله انتساب اعتبار به کار برده می‌شود. هر وقت عامل پاداش می‌گیرد. این پاداش باعث افزایش قدرت کنشهایی می‌شود که در همان لحظه اجرا شده‌اند. بدین ترتیب آن دسته از طبقه‌بندهایی که در طول زمان باعث رسیدن به پاداش می‌شوند، قوی‌تر شده و بنابراین بیشتر اجرا می‌شوند.

۳- **استراتژی استخراج:** تعداد طبقه‌بندها ثابت است. هر از چندی، عامل طبقه‌بندِ دارای کمترین قدرت را حذف کرده و به جای آن ترکیبی از طبقه‌بندهای پرقدرت جایگزین می‌سازد. با این روش، عامل سعی در استخراج روشهای دیگر می کند، در حالیکه این استخراجها حول و حوش جوابهای مطلوب هستند.

یکی از جنبه‌های جالب طبقه‌بندها استراتژی تجربه آنها است که پیچیده‌تر از یادگیری پاداشی است. ایدهٔ اصلی آن این است که جواب بهتر برای یک مسأله با تغییر اندک جواب خوب و یا با ترکیب جوابهای خوب بدست می‌آید. متأسفانه سیستم طبقه‌بند برخی از محدودیتهای عاملهای پاداشی را داراست. به ویژه مسألهٔ اهداف متغیر نسبت به زمان و یادگیری در پایان تسلسل را دارا است. از آنجا که سیستم حافظه ندارد ممکن است بارها یک طبقه بند را ارزیابی کند مثلاً ممکن است یک طبقه‌بند را به خاطر مقدار قدرت کمش حذف کند ولی بلافاصله آنرا بسازد. که چه چیزهایی را حذف کرده است. از دیگر سوی، این حقیقت که سیستم طبقه‌بندهای کم قدرت را حذف می‌کند باعث افزایش کارآیی در زمان انتخاب کنش می‌گردد.

۳-۵-۳-روش سازنده مدل

دستهٔ آخر روشهای یادگیری عاملها به جای یادگیری نگاشت وضعیت-کنش، سعی در یادگیری‌رابطهٔ سببی[۱] کنشها می‌کند. مدل Drescher که بر اساس نظریه‌هایی در مورد رشد کودک است، نمونه‌ای پیچیده از این سیستم‌هاست[٦]. عامل یک مدل احتمالاتی از تأثیرات انتخاب کنشها در وضعیت درست می‌کند. از این مدل سببی برای انتخاب کنش استفاده می‌شود. انتخاب کنش و یادگیری در این روش از هم مجزا هستند.

در بیشتر عاملهایی که از این روش استفاده می‌کنند، عامل نگاشت کامل بین تمام زوجهای (وضعیت – کنش و وضعیت جدید) را یاد نمی‌گیرد. بلکه عامل نگاشت بین جزئی از وضعیتها را به جزیی دیگر از وضعیت یاد می‌گیرد. عامل آن دسته از جنبه‌های مهم وضعیت (داده حسگر آنها شرط لازم و کافی هستند) را به انضمام کنش به جنبه‌های مهم وضعیت جدید (داده‌هایی از حسگرهاا که تغییر کرده‌اند)نگاشت می کند. به تلفیق مجموعهٔ شرایط، کنش و مجموعه‌ای از مقادیر مورد انتظار، "شما"، "ماژول" یا "رفتار" گفته می‌شود.

این روش سه مسأله طراحی به صورت زیر حل کرده است.

۱- **مکانیزم انتخاب کنش:** عاملی که با این روش ساخته می‌شود قادر به حل مسأله هدفهای متغیر یا چندهدفی چندگانه هستند. با داشتن اهداف، این عاملها در زمان کار "شما"هایی که دارای بیشترین ارتباط

---

[۱] Causual

با هدف و قابلیت اعتماد بیشتری هستند را برمی‌گزیند. علاوه بر یادگیری، فرآیند انتساب مقدار اعتبار نیز وجود دارد. اغلب این مقدار نشان دهندهٔ ماژولهایی است که نشان داده‌اند قابلیت اعتماد بیشتری دارند. این مقدار ممکن است تعداد کنشهای موجود در تسلسلهایی باشد که منتهی به هدف شده اند.

۲- **روش یادگیری:** هر وقت کنشی انجام شد، عامل تغییرات بوجود آمده در محیط را بررسی می کند. از این اطلاعات برای یادگیری همبستگی بین زوج وضعیت – کنش و نتایج استفاده می‌کند. پس از انجام هر کنش تمام شِماها با اضافه شدن مثال جدید، بروز می‌شوند. گاهی لازم می‌شود عامل از شماهای موجود شمای جدیدی بوجود آورد. عامل قادر به تشخیص نیاز شِما به شرطهای بیشتر به منظور افزایش قابلیت اعتماد، می‌باشد. و بدین منظور نسخه‌هایی از شِماها که دارای شروط بیشتری هستند، را بوجود خواهد آورد.

۳- **استراتژی استخراج:** استراتژی‌های متفاوتی برای استخراج موجود است. مثلاً در سیستم Drescher با وجود به کارگیری یادگیری پیچیده از روشی بسیار ساده برای استخراج استفاده می‌کند. عامل او با انجام آزمایش‌های تصادفی یاد می‌گیرد. Foner در [۸] روشهایی را بیان کرده که روش تصادفیDrescher را بهبود بخشیده و آنرا هوشمندتر و با هدف مرتبط‌تر می‌کند.

یکی از اهداف مهم این روش، آن است که می‌تواند رفتارهای یاد گرفته شده در مورد یک هدف را در مورد هدف دیگر استفاده کند. بنابراین سیستم بر اساس چگونگی تأثیر یک کنش در یک وضعیت و وضعیت حاصله، مدلی را می‌سازد که راه نیل به مجموعه‌ای از اهداف را نشان می‌دهد. علاوه بر آن در هر کنش عمل یادگیری را انجام می‌دهد (برخلاف دو روش قبل). مزیت دیگر این روش، آن است که طراح می‌تواند به راحتی دانش دامنه را به سیستم دهد و سیستم از آن استفاده کند. عامل حتی می‌تواند این دانش را در صورت غلط بودن تصحیح نماید.

عیب این روش تأخیر در انتخاب کنش است زیرا نگاشت مستقیم بین وضعیتها به کنشهای بهینه وجود ندارد.

بزرگ شدن محیط مسأله مشکل تمام روشهای فوق است زیرا پیچیدگی الگوریتمهای آنها از آن است که بتوان در مسائل حقیقی و پیچیده از آن‌ها استفاده کرد.

٤-١- مقدمه

سیستمهای مبتنی برعامل روش جدیدی برای توسعه نرم افزار هستند. کاربرد عاملها بصورت روز افزون افزایش یافته است. این کاربردها درابعاد مختلفی از سیستمهای کوچک مانند فیلتر کنندهٔ نامه های الکترونیکی گرفته تا سیستمهای بزرگ، باز و حساس مانند کنترل ترافیک هوایی استفاده می شود.

Jennings در تعریف سیستمهای مبتنی بر عامل می گوید:"سیستم مبتنی بر عامل آن دسته از سیستمهایی هستند که انتزاع آنها در سطح عامل است"[۱۳]. وی همچنین موارد استفاده از عامل را دو مورد زیر بر می شمرد :

۱- مسائلی که از حد اتوماسیون فراتر بوده و تکنولوژی آن موجود نمی باشد.

۲- راه حل مبتنی بر عامل ، درک مسأله و تهایتاً حل آن را ساده تر و کارآتر سازد.

Nwana مواردی که سیستم مبتنی بر عامل واقعا تنها راه حل نیستند را بدینگونه بیان می کند[۲۱].

۱- زمانیکه عامل تنها به منظور، ارتقاء ماژولاریتی(با کم کردن پیچیدگی)، سرعت(با توازی)، قابلیت اطمینان (با افزونگی)، کارآیی و انعطاف پذیری استفاده شود.

۲- زمانیکه سیستم آنقدر ساده باشد که یک عامل به تنهایی قادر به انجام وظیفهٔ سیستم باشد.

۳- زمانیکه سیستم چند عاملی تنها به منظور فائق آمدن بر استنتاجات ناهمگون استفاده شود. در حالیکه از سیستمهای Blackboard نیز می توان استفاده کرد.

در ادامه، با مبنا قرار دادن نظرJennings در[۱۳] به کاربردهای مهم عاملها می پردازیم.

٤-٢- حل مسائل جدید

موارد خاصی از سیستمهای نرم افزاری پیچیده تر از آن هستند که بتوان آنها را به درستی طراحی و پیاده سازی کرد. ساده ترین نوع سیستمهای نرم افزاری سیستمهای تابعی[۱] می باشند. این گونه سیستمها بر روی ورودی پردازش خاصی انجام داده و خروجی تولید می کنند. درمقابل این سیستمها، سیستمهای واکنشی هستند. این سیستمها با محیط تعامل داشته و در برابر تغییرات مختلف محیط واکنش نشان می دهند، مثلاً، سیستمهای کنترل فرآیند، سیستمهای عامل و سیستمهای مدیریت شبکه.

در این نوع کاربردها، سیستم باید برای مدت طولانی در محیطی پویا کار کند. به همین خاطر زبانها، متدولوژی ها و ابزار متعددی برای غلبه بر پیچیدگی سیستم بوجود آمده، با این وجود هنوز در برخی از کاربردهای روشها و ابزار مهندسی نرم افزار ناتوان است و روشهای جدیدی مورد نیاز است. این نوع سیستمها عبارتند از:

۱- سیستمهای باز

۲- سیستمهای پیچیده

۳- سیستمهای کامپیوتری فراگیر

٤-٢-١- سیستمهای باز

سیستمهایی هستند که ساختار آنها می تواند بصورت پویا تغییر کند.یعنی اجزا آنها مشخص نبوده و در طول زمان تغییر می کند و حتی ممکن است کاملاً ناهمگن هم باشند.

---

[۱] Functional

این اجزا توسط افراد مختلف ، در زمانهای مختلف، با ابزارها و تکنیکهای مختلف بوجود می آید. اینترنت مثالی از سیستم نرم افزاری باز است. هر گره اینترنت دارای ساختار و هدفی خاص است و سیستم نرم افزاری باید قادر به تعامل با این ساختارها و اهداف، بدون راهنمایی از سوی کاربر، باشد.

٤-٢-٢- سیستمهای پیچیده

قویترین ابزار مقابله با پیچیدگی سیستم ماژولاریتی و انتزاع می باشند و عامل، ابزاری قوی برای ماژولاریتی است. در چنین سیستمهایی مسألهٔ کلی به مسائل کوچکتر تقسیم می شود و هر عامل مسألهٔ خود را حل می کند. این تقسیم بندی اجازه می دهد که هر عامل بهترین راه حل را برای حل مساله خود انتخاب کند.
علاوه بر آن ، عامل انتزاع خوبی را نیز فراهم می آورد. به طوریکه به طراح اجازه می دهد سیستم پیچیده نرم افزاری را به عنوان جامعه ای از عاملهای حلال مسأله تصورکند.

٤-٢-٣- سیستمهای کامپیوتری فراگیر

کاربران هنوز در استفاده از واسط کاربری مشکل دارند. باید کامپیوترها نقش مهمتری در ارتباط با کاربران اجرا کنند. Negropote بر این عقیده است که در آینده کاربران از کامپیوترها به عنوان وکیل خود در انجام کارها استفاده خواهند کرد[٢٠]. جهت نیل به این ایده باید نرم افزار دارای چهار ویژگی باشند.آن چهار ویژگی عبارتند از:

١- خودگردانی: باید بتواند بدون راهنمایی کاربر مساله را به بهترین شکل حل نماید.
٢- کنش گرایی: نباید منتظر شود تا کاربر کار آن را مشخص کند، بلکه او خود باید به کمک کاربر بشتابد.
٣- پاسخگویی: باید بتواند با تغییر خواسته های کاربر وظایف خود را تغییر دهد.
٤- انعطاف: باید توانایی تشخیص علایق کاربر را داشته و در تعامل با وی این موضوع را نشان دهد.

٤-٣-افزایش کارآیی

عاملها نه تنها درساخت کاربردهایی که قبلاً قادر به تولید آن نبوده ایم، ما را یاری می کنند. بلکه برای بعضی از کاربردهای فعلی سطح انتزاع بهتری می سازند. کاربردهایی که برای تبدیل به کاربردهای مبتنی بر عامل مناسب می باشند، عبارتند از:

١- داده ، کنترل ،تجربه یا منابع ذاتاً توزیع شده باشند.
٢- سیستم ذاتاً مجموعه ای از اجزای خودگردان همکار باشد.
٣- سیستم دارای ماژولهای قدیمی باشد که باید به گونه ای ساخته شود تا با دیگران ارتباط برقرار کند.

٤-٣-١-داده ، کنترل ،تجربه یا منابع ذاتا توزیع شده

زمانیکه دامنهٔ مسأله شامل تعدادی حلال مسأله یا منابع توزیع شده بطورمنطقی یا فیزیکی باشد و همچنین به تعامل با یکدیگر به منظور حل مساله نیاز داشته باشند. در این موارد، اغلب استفاده از عاملها جوابهای مطلوبی ارائه می دهند. مثلاً در سیستم پزشکی توزیع شده، پزشکان عمومی ، متخصصان بیمارستان، پرستاران و نهادهای بهداشتی خانواده باید با همدیگر در جهت مراقبت از مریض گام بردارند. در این دامنه:

١- توزیع داده : داده هایی که پزشکان عمومی از بیماردارند با پرستاران بیمارستان متفاوت است.
٢- توزیع کنترل: هر کس باید وظیفهٔ خاص خود را انجام دهد.

۳- توزیع تجربه: دانش یک پزشک با سایر پزشکان یا پرستاران متفاوت است.
٤- توزیع منابع: یکی در مورد کارهای تخت بیمار ، یکی درمورد پرداختهای بیمار و .... کار می کند.

٤-۳-۲- سیستم ذاتاً مجموعه ای از اجزای خودگردان همکار باشد.

غالباً عامل خودگردان معادل (متافور) کارهای نرم افزار است. مثلا، برنامه ای که نامه های الکترونیکی را فیلتر می کند را می توان با معادل آن PDA[۱] بیان کرد. زمان بند ملاقاتها طبیعتا یک عامل قوی، خودکار و اجتماعی است که می تواند با عاملهای دیگر به جای کاربر خود ارتباط برقرار کند.

Batesو Wavish [۱] بر این باورند که شخصیتها در بازیهای کامپیوتری وحقیقت مجازی می توانند عامل باشند

٤-۳-۳- سیستمهای قدیمی

سازمانها، نرم افزارهای زیادی دارند که با تغییرخواسته ها باید آنها تغییر کند. اما این گونه تغییرات همواره مشکل زا می باشند. زیرا باعث خرابی برخی از اجزای سیستم ، مستندات و ... می شوند. تنها راه آن است که آنرا به قسمتهای مجزا تقسیم کنیم و تا با استفاده از عاملهای لفاف[۲]با سیستمهای جدید دیگر تعامل کند. به این فرآیند را "عاملی کردن" گویند.

٤-٤- دامنه کاربرد عاملها

عاملها دارای ابعاد مختلفی هستند که بر حسب آن می توان کاربردهای آنها را دسته بندی کرد. این ابعاد می توانند نوع عاملها ، تکنولوژی استفاده شده برای پیاده سازی و یا حتی دامنه کاربرد باشد. در این بخش برخی از کاربردهای عوامل برحسب دامنه کاربردها آمده است.

٤-٤-۱- کاربردهای صنعتی

کاربردهای صنعتی از اولین کاربردهای عاملها بودند. در سال ۱۹۸۷ ، Parunak تجربهٔ خود را در استفاده از عاملها در موردتخصیص وظیفه درمحیط کارخانه ای اعلام کرد[۲۲]. برخی از کاربردها از این دست عبارتند از :

٤-٤-۱-۱- کنترل فرآیند[۳]

کنترل فرآیند، کاربرد طبیعی عاملهای هوشمند است. چون کنترلگرهای صنعتی سیستمهای واکنشی خودگردان هستند. سیستم مدیریت توزیع برق که با استفاده از ARCHON پیاده سازی شده و در اسپانیا در حال استفاده است و یا کنترل شتاب دهنده ذرات[٤] نمو نه هایی از این کاربرد هستند.

---

[۱] Personal Digital Assistant

[۲] Wrapper

[۳] Process Control

[٤] Particle Accelerator Control

٤-٤-١-٢-تولید

YAMS[1]نمونه ای از استفادهٔ عامل در این کاربردها است. هدف YAMS مدیریت بهینه فرآیند تولید است.YAMS مسأله را بصورت مذاکره بین اجزا می بیند. این سیستم در این تلاش است که دستورات تولید را بطور کارآ به کارخانه ها و واحدهای تولیدی تخصیص دهد.

٤-٤-١-٣-کنترل ترافیک هوایی

کنترل ترافیک هوایی یکی از کاربردهای مهم عاملها است. OASIS در فرودگاه سیدنی به کار می رود. در این سیستم عاملها نقش هواپیما و سیستمهای کنترلی را دارند. زمانیکه هواپیما می نشیند یک عامل به آن تخصیص داده می شود. OASIS با استفاده از ابزار dMARS پیاده سازی شده که معماری عاملهای برنامه ریز را دارد.

٤-٤-٢-کاربردهای تجاری

٤-٤-٢-١-مدیریت اطلاعات

مدیریت اطلاعات شامل دو بخش است، فیلتر کردن اطلاعات و جمع آوری اطلاعات. در جهت نیل به هر یک عاملهایی درست شده اند. ازجملهٔ آنها می توان به موارد زیر اشاره کرد:

Maxim: کاری از مایز است که فیلتر کردن نامه های الکترونیکی را بر عهده دارد. Maxim طبقه بندی ، حذف، ارجاع و بایگانی نامه ها را بر اساس رفتار کاربر یاد می گیرد.

Newt : سعی در فیلتر کردن اینترنت دارد.Newt به زبان C++ بوده و روی یونیکس پیاده سازی شده است. ورودی آن اخبار گروه های خبری و خروجی مقاله مدون آن است.

کتابخانه دیجیتال **Zuno** :اجازه می دهد کاربر دیدی منسجم از مجموعه داده های نامنسجم و سازمان نیافته داشته باشد. اینترنت نمونه ای از اینگونه داده هاست.

٤-٤-٢-٢-تجارت الکترونیکی:

تجارت از تعامل بین انسانها بوجود می آید و می توان با داشتن ابزار تصمیم گیری مناسب آن را اتوماتیک کرد[٤]. Kasbeh یک مثال خوب از سیستمی است که سعی داشته که تجارت را بطور اتوماتیک بین عاملهای کاربران انجام دهد.

٤-٤-٢-٣-مدیریت فرآیند تجارت:

مدیران نیازمند اطلاعات درست و بروز از واحدهای مختلف برای تصمیم گیری صحیح می باشند. سیستمهای اطلاعاتی زیادی بدین منظور تهیه شده اند. از عاملها نیز در این راستا می توان استفاده کرد. سیستم ADEPT یک نمونه از این سیستمها است. ADEPT سعی داشته مدیریت فرآیند تجارت را با استفاده از دو عامل مذاکره کننده و سرویس دهنده انجام دهد. این سیستم در British Telecom در حال استفاده است.

---

[1] Yet Another Manufacturing System

٤-٤-٣-کاربردهای پزشکی

٤-٤-٣-١-بررسی کردن وضعیت مریض

Guardian سیستمی است که برای مدیریت فوریتهای پزشکی ساخته شده است. این سیستم چند عاملی ویژگیهای شایان ذکر زیر را داراست:

١- عاملها هر یک تجربه منحصر به خود را دارند.

٢- عاملها در گروه به خوبی اطلاعات را به اشتراک می گذارند.

٤-٤-٣-٢-مواظبت از سلامت:

Huang یک نمونه اولیه با تعداد زیادی عامل از جمله عاملهای پزشک، پرستار، کارمندان دفتری بیمارستان را ارائه داده است که وظیفهٔ مراقبت از سلامت بیمار را انجام می دهد[١١].

٤-٤-٤-تفریح

بازیهای کامپیوتری و حقیقت مجازی از جمله کاربردهای استفاده عاملها در جنبه تفریح و سرگرمی کامپیوتری می باشند. در حقیقت می توان به جای هر شخصیت موجود در بازی یک عامل قرار داد. بدین صورت، بازی تعامل بین عاملها ست.

٤-٥- عاملها و کاربردهای تجاری

هر چند که عامل یکی از مباحث داغ نرم افزار است اما هنوز محدود به تحقیقات است. برخی از شرکتهای بزرگ مانند IBM به دنبال عاملی هوشمند است که بتواند در تجارت الکترونیکی کاربرد داشته باشد. Microsoft تعبیری ساده از عامل دارد و گاه به برنامه های مقیم حافظهٔ خود عامل اطلاق می کند. برخی از شرکتها از عامل به عنوان رویکردی برای نرم افزار منعطف استفاده می کنند.

اما در مدت انجام این تحقیق عاملی که جنبهٔ تجاری پیدا کرده باشد، یافت نشد.

منابع و مآخذ

[1] J. Bates. **Virtual Reality, art, and entertainment**. PRESENCE: Teleoperators and Virtual Environments, 1992.

[2] R. A. Brooks. **Intelligence without reason**. Proc. IJCAI-91, Morgan Kaufmann, 1991.

[3] Jose C , Brustoloni. **Autonomous Agents: Characterization and Requirements**. Carnegie Mellon Technical Report CMU-CS-91-204, Pittsburgh: Carnegie Mellon University, 1991.

[4] A. Chavez, P. Maes. **Kasabe: An Agent marketplace for buying and selling goods**. Proc. Of First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Systems, London, UK,1996.

[5] D.C. Dennet. **The Intentional Stance**. The MIT Press, 1987.

[6] G. L. Drescher. **Made-Up Minds: A constructivist Approach to Artificial Intelligence**, MIT Press, 1991

[7] M. Fisher. **A survey of concurrent MetaM – The language and its applications**. In D. Gabby and H. J. Ohlbach, editors, Proceeding of First International Conference on Temporal Logic (ICTL-94). Springer-Verlag, July 1994.

[8] L. Foner . **Paying Attention to What's Important: Using Focus of Attention to Improve Unsupervised Leaning**. Third Int. Conference on Simulation of Adaptive Behaviour, Brighton, August 1994.

[9] B. Hayes-Roth. **An Architecture for Adaptive Intelligent Systems**. *Artificial Intelligence:* Special Issue on Agents and Interactivity, 72, 329-365, .1995.

[10] C. Hewitt. **Viewing control structures as patterns of passing messages**. Artificial Intelligence, 1977.

[11] J. Huang, N. R. Jennings, J. Fox. **An agent-based approach to health care management**. Int. Journal of Applied Artificial Intelligence.1995.

[12] N. R. Jennings and M. Wooldridge. **Applications of Agent Technology**. In N. R. Jennings and M. Wooldridge, editors, *Agent Technology: Foundations, Applications, and Markets*. Springer-Verlag, March 1998.

[13] N. R. Jennings and M. J. Wooldridge. **Applications of Intelligent Agents** in Jennings, N. R. and Wooldridge, M. J., Eds. , pages 3-28 ,1998.

[14] N. R. Jennings and M. Wooldridge. **Software Agents**. *IEE Review* 42(1), pages 17-21. January 1996.

[16] L. P. Kaelbling. **An architecture for intelligent reactive systems** In M. P. Georgeff (ed) Reasoning About Actions & Plans-Proc. Of 1986 Workshop,Morgan Kaufmann Publishers, 1986.

[17] P. Maes. **Modeling adaptive autonomous agents**. In C. G. Langton, editor, Artificial Life, An Overview, Cambridge, Massachusetts, 1995. MIT Press.

[18] P. Maes, editor. **Designing Autonomous Agents**. The MIT Press,1990.

[19] J. McCarthy. **Ascribing mental qualities to machines**. Technical Report, Stanford AI Lab.

[20] N. Negroponte. **Being Digital**. Hodder and Stoughton, 1995.
[21] H. S. Nwana and M. Wooldridge. **Software Agent Technologies**. In *BT Technology Journal* 14(4), pages 68-78, October 1996.
[22] H. V. D. Parunak . **Manufacturing experience with the contract net**. In M. N. Huhns (Ed.) Distributed AI. Morgan Kaufmann,1987.

[23] Stuart J Russell and Peter Norvig , **Artificial Intelligence: A Modern Approach**, Englewood Cliffs, NJ: Prentice Hall.,1995.

[24] N. Seel. **Agent Theories and Architectures**. PhD Thesis, Surrey University, Guildford, UK, 1989.

[25] N. Shardlow. **Action and agency in cognitive science**. Master's thesis, Department of Psychology, University of Manchester.

[26] Y. Shoham. **Agent-oriented programming**. Artificial Intelligence, 60(1): 51-92, 1993.

[27] H. Simon. **The Science of the Artificial**, MIT Press, 1969.

[28] D. C. Smith , A. Cypher and J. Spohrer. **KidSim: Programming Agents Without a Programming Language.** *Communications of the ACM*, 37, 7, 55-67, 1994.

[29] P. Todd and S. Wilson. **Environment Structure and Adaptive Behaviour from the Ground Up**. Proc. 2nd Int. Conference on Simulation of Adaptive Behaviour, 1995.

[30] S. R. Thomas. **PLACA, an Agent Oriented Programming Language**. PhD Thesis, Computer Science Department, Stanford University, August 1993.

[31] M. Wooldridge and N. R. Jennings. **Intelligent Agents: Theory and Practice**. In *Knowledge Engineering Review* 10(2), 1995.

[32]A. Yonezawa, editor. **ABCL- An Object Oriented Concurrent system**. The MIT Press, 1990.